Einführung in die Angewandte Bioinformatik: Begleitende Einführung in R 15.04.2009 – 14.05.2009

Prof. Dr. Sven Rahmann

Zusatzangebot: Hilfe zu R (Keine Verpf ichtung!)

Markus Kemmerling (SHK aus Studienbeiträgen)
Wolfgang Walz (SHK)

Zeit Mi nachmittags in OH14, Terminalpool 2.Etage
Ort OH14, Terminalpool 2.Etage
oder angebio@lists.cs.tu-dortmund.de (Mailingliste)

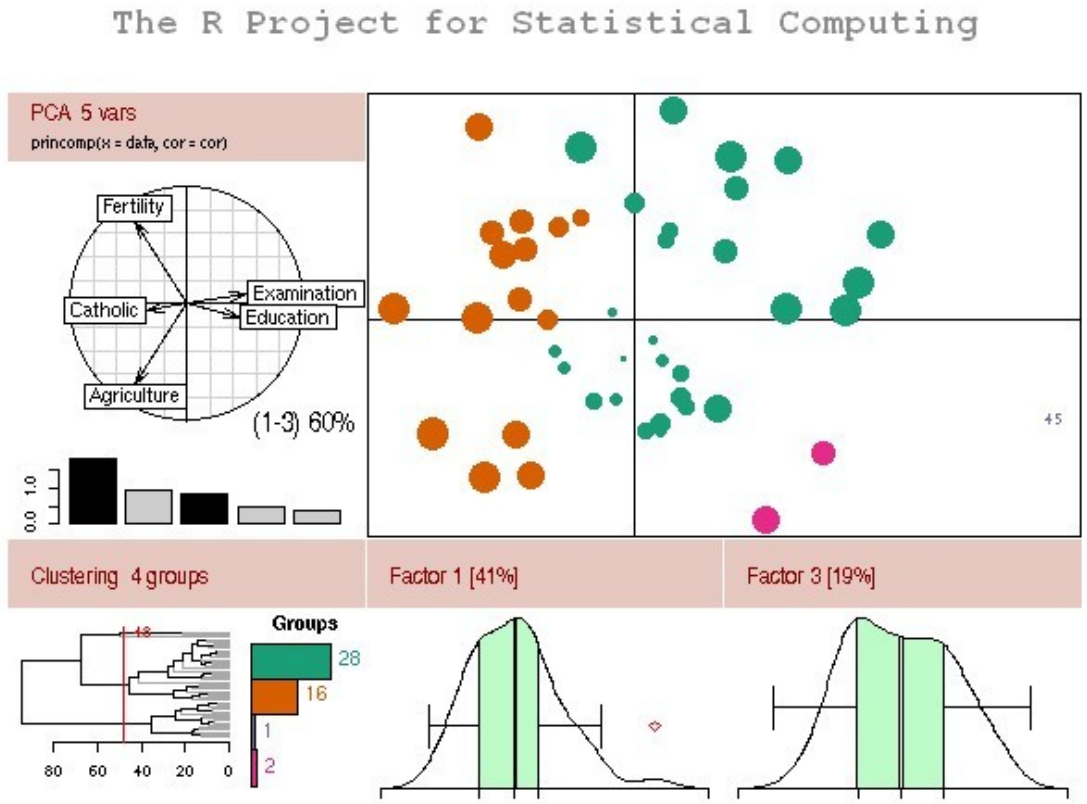
Alle Informationen

Webseite zur Vorlesung: http://bioinfo.wikidot.com/

Sprechstunde von Prof. Sven Rahmann

Mo 16-17 in OH14, R214
Bitte möglichst per e-mail anmelden, sonst evtl. sehr lange Wartezeiten!
Sven.Rahmann /at/ tu-dortmund.de

Statistische Datenanalyse mit R: http://www.r-project.org



About R
What is R?
Contributors
Screenshots
What's new?

Download
CRAN

R Project
Foundation
Members & Donors
Mailing Lists
Bug Tracking
Developer Page
Conferences
Search

Documentation
Manuals
FAQs
Newsletter
Wiki

Getting Started:

- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To download R, please choose your preferred CRAN mirror.
- If you have questions about R like how to download and install the software, or what the license terms are, please read our

Über R

R ist eine freie Software zur statistischen Datenanalyse
(free software – free as in „beer“ and as in „speech“).

Wir besprechen einige statistische Grundlagen,
und wie man R zur einfachen Datenanalyse einsetzt.

Auf den Rechnern im Übungspool ist R vorinstalliert.
Zu Hause laden Sie sich R umsonst unter [http://www.r-project.org] herunter.

Erste Schritte:

- Starten und Verlassen von R
- R als Taschenrechner
- R als Plotter

Starten und Verlassen von R; Hilfe

Starten von R

Im Übungs-Pool: Eingabe des Kommandos „R“ in der Shell
Unter Windows: Herunterladen und Aufrufen des Installers ->Desktop-Icon

Hilfe in R

R funktioniert wie die Shell: Befehle werden nacheinander abgearbeitet
(aber es stehen andere Befehle zur Verfügung als in der Shell!)
Hilfe bekommt man mit `help()` oder einem vorangestellten Fragezeichen `?`

Verlassen von R

Aufruf von `q()`
q ist eine Funktion, daher wird sie mit Klammern aufgerufen.
In den Klammern stehen ggf. Argumente, aber q hat keine.

R als Taschenrechner

Es stehen alle arithmetischen Operationen,
elementaren mathematischen Funktionen,
sowie logischen Operationen zur Verfügung.

Die Eingabe eines Ausdrucks liefert dessen Wert (Ergebnis) zurück.
Ergebnisse können Variablen zugewiesen werden,
um sie für die Dauer der R-Sitzung zu speichern.

```
(5 + 3) * 17 / 2 ** 2
x = (5 + 3) * 17 / 2 ** 2
x
x + 6
?sin # Hilfe zur sin-Funktion
sin(tan(x))
```

Wissenschaftliche Notation von Zahlen

Große und kleine Zahlen werden oft in der Exponentialnotation angegeben.

- $107 = 1.07 * 10^{2} = 1.07E+02$
- $0.334 = 3.34 * 10^{-2} = 3.34E-02$

Das E steht für Exponent (zur Basis 10), es sollte ein großes E sein;
leider sieht man bisweilen auch ein kleines e.

Dieses E oder e hat nichts mit der Zahl e (etwa 2.71) zu tun!

Logik in R

Logik in R ist 3-wertig:
- wahr (T, TRUE)
- falsch (F, FALSE)
- nicht entscheidbar / fehlende Daten (NA)

```
x == 34
x > 40
(x > 30) & (x < 40)
(0 / 0) > 2
```

Aus fast allen Berechnungen, in die NA hineingesteckt wird, kommt wieder NA heraus. Vergleich mit NA funktioniert nicht! Statt dessen muss die Funktion `is.na()` verwendet werden.

```
y = ((0 / 0) > 2)    # y ist jetzt NA
y == NA              # liefert nicht TRUE, sondern wieder NA
is.na(y)             # liefert TRUE
```

Vektoren

Häufig soll dieselbe Operation auf mehrere Werte angewendet werden.
Dies kann man erreichen, wenn die Daten in einem Vektor gespeichert sind.
Erzeugen eines Vektors mit c() durch Aufzählung oder mit „:“ (von-bis).

```
y = c(2,3,5,7,11,NA,13)
z = 1:10

y * 2
z + 3
```

Man kann zwei Vektoren element-weise miteinander verknüpfen.
Dabei wird der kürzere solange wiederholt, bis die Länge des längeren erreicht ist.
Man erhält eine Warnung, wenn die Längen keine Vielfachen voneinander sind.

```
y + z
```

Vektoren mit seq()

Regelmäßig strukturierte Vektoren erzeugt man mit der seq()-Funkton durch Angabe von Startwert, Endwert, Schrittweite („by"):

```
x = seq(0, 10, by=0.5) # erzeugt 0, 0.5, 1, 1.5, ..., 9.5, 10
y = sin(x)
```

Zugriff auf Elemente von Vektoren

Vektoren werden mit eckigen Klammern [] indiziert.
Indizierung beginnt bei 1, nicht bei 0 !
Man kann auch mit Vektoren indizieren!

```
x = seq(0, 10, by=0.5)      # erzeugt 0, 0.5, 1, ..., 9.5, 10
x[0]                        # gibt es nicht
x[1]                        # 0
x[length(x)]                # 10
x[seq(1,length(x),3)]       # jedes dritte Element von x
```

Tests von Vektoren, logische Indizierung

Elemente von Vektoren können auf Eigenschaften getestet werden.
Wie viele von 20 gleichverteilten Zufallszahlen auf [0,1] sind >0.5 ?

```
x = runif(20)          # 20 Zufallszahlen zwischen 0 und 1
gr = x>0.5             # Vektor aus TRUE und FALSE
sum(gr)                # Anzahl der TRUE-Werte
y = x[gr]              # x-Werte >0.5 nach y extrahieren
mean(y)                # Durchschnitt dieser Werte
mean(x[x>0.5])         # dasselbe! Lies:
                       # Mittelwert der x-Werte, für die x>0.5
```

Merke:
Vektor-Indizierung kann auf zwei Arten erfolgen:
- numerisch (mittels Zahlenvektor: x[c(1,2,3)])
- logisch (mittels T/F-Vektor: x[c(T,T,T,F,F,F)])

Einfache Plots

Die graphischen Fähigkeiten von R sind sehr mächtig.
Beispielsweise kann man leicht einfache Funktionsplots erstellen,
z.B. von $y = \sin(x^2)$ auf dem Intervall [-5, 5]:

```
x = seq(-5, 5, by=1/16) # kurze Schrittweite für schönen plot
y = sin(x**2)
plot(x,y)
```

Der resultierende Punktplot sieht nicht sehr übersichtlich aus.
Besser, wir hätten eine durchgehende Linie statt einzelner Punkte.

```
plot(x,y, type="l")
plot(x,y, type="o")  # Punkte und Linien übereinander
```

Oder in Blau mit schönem Titel:

```
plot(x,y, type="o", col="blue", main="Parabel")
```

Funktionsaufruf mit benannten Parametern

Es ist Konvention, einer Funktion erst die nötigen Daten zu übergeben; danach benannte Parameter, die das Verhalten der Funktion verändern.

Beispiel: Plot mit Linien und Punkten in Blau mit Titel:

```
plot(x,y, type="o", col="blue", main="Ein Funktionsplot")
```

Beispiel: Logarithmische Achsen
Achsen können mit logarithmischer Skala darzustellen.
Dies geschieht (für die y-Achse) durch Angabe des Parameters log="y".
Monome der Form $y=cx^n$ werden bei logarithmischen Achsen zu Geraden.

```
x = seq(1, 5, by=1/16)
y = 5 * x**3
plot(x,y, type="o")
plot(x,y, type="o", log="xy")  # beide Achsen logarithmisch
```

Beschreibende Statistik mit R

Sei v ein Vektor von reellen Zahlen.
Wir definieren und berechnen einige Kennzahlen.

Mittelwert	mean(v)	Durchschnitt
Varianz	var(v)	Mittlere quadrat. Abw. zum MW
Standardabweichung	sd(v)	Wurzel aus Varianz
Median	median(v)	50% der Werte kleiner, 50% größer
p-Quantil	quantile(v,p)	Anteil *p* der Werte kleiner; 1-*p* größer
p-Quantile	quantile(v,p)	*p* kann Vektor sein!
Interquartilabstand	IQR(v)	75%-Quantil – 25%-Quantil

Beschreibende Statistik mit R

5 Kennzahlen fivenum(v) # dazu ?in R fivenum lesen

- Mimimum (oder mindestens Median – 1.5*IQR)
- 25%-Quantil
- Median
- 75%-Quantil
- Maximum (oder höchstens Median + 1.5* IQR)

Boxplot boxplot(v) # Visualisierung von fivenum

Histogramm hist(v) # Anzahl der Zellen sinnvoll gewählt
hist(v,*n*) # Histogramm mit *n* Zellen

Ein Histogramm zeigt zu jedem Wert(ebereich) die Anzahl der Elemente in v an, die in diesen Bereich fallen.

Tabellen (data frames) einlesen

Wie kommen die Daten in den R-Workspace ?
- Manuelle Eingabe: langsam, umständlich
- Einlesen aus Datei: schnell, aber Format muss beachtet werden

Aktuelles Verzeichnis anzeigen / wechseln (wie `cd` in der Shell):
`getwd()` / `setwd("neuesVerzeichnis")`

Tabelle aus Datei in Variable x einlesen:
```
x = read.table(dateiname, ...)
```

Optionen dabei z.B.:
`sep = ""` oder `sep=","` oder `sep=";"` (Trennungszeichen der Elemente)
`header = FALSE` oder `header=TRUE` (erste Zeile enthält Namen?)
`col.names = vektor` (Spaltennamen explizit angeben)

Mit Tabellen (data frames) arbeiten

Eingelesen Daten `x` stehen in sog. **data frames** (Datenrahmen = Tabellen).
Jede Zeile ist ein Datensatz; jede Spalte repräsentiert ein Attribut der Daten.

Größe anzeigen lassen: `dim(x)`
Spalten haben Namen (aus Datei oder automatisch): `colnames(x)`
Spalten umbenennen: z.B. `colnames(x) = c("Anna", "Bert")`
Kurzübersicht: `str(x)`

Zugriff auf Zeile 17: `x[17,]` (Komma beachten!)
Zugriff auf mehrere Zeilen 17 bis 23: `x[17:23,]` (Komma beachten!)
Zugriff auf Spalte "Anna" als Vektor: `x$Anna` oder `x[,"Anna"]`
Auf Spalten direkt (ohne Präfix `x$`) zugreifen: `attach(x)`

Tabelle wieder speichern: `write.table(x, Dateiname, ...)`

Erste statistische Analysen mit R

Vergleich von zwei Vektoren x,y gleicher Länge

Scatterplot	plot(x,y)	Punkte (x[i],y[i])
Korrelationskoeffizient	cor(x,y)	+–1 bei linearer Abhängigkeit 0 bei Unabhängigkeit

Komplexes Beispiel einer beschreibenden Datenanalyse

Mit einer neuen Sequenziertechnologie (ABI SOLiD) wurden
kurze nicht-codierende RNA-Stücke (ncRNA reads) sequenziert.

Gegeben

- FASTA-Datei mit ca. 670000 Sequenzen der Länge 35.
- FASTA-ähnliche Datei mit Qualitätswerten (-10*log10(Fehlerwahrscheinlichkeit))
  ebenfalls ca. 670000 Sequenzen à 35 Werte (.qual-Datei)

Fragestellung

Betrachte nur die Qualitätswerte, nicht die Sequenzen.
Nimmt die Qualität „hinten" ab?
Sind Qualitätswerte an Positionen 30+ niedriger als an den ersten Positionen?

Erste Schritte – Daten anschauen

Gegeben

FASTA-ähnliche Datei mit ca 670000 x 35 Qualitätswerten (.qual-Datei)

In der Shell

```
% ls -l reads.qual          # wie groß ist die Datei?
% head -n 30 reads.qual     # erste 30 Zeilen
% tail -n 30 reads.qual     # letzte 30 Zeilen
% cat reads.qual            # ganze Datei (Unsinn!)
% more reads.qual           # durchblättern
% less reads.qual           # wie more
% wc less.qual              # Wie viele Zeilen?
```

FASTA-ähnliche Datei mit Qualitätswerten

```
...
# Title: s0329_20090331_552to561_613to614_2_552_561
>854_648_594_F3
25 27 27 2 29 30 25 3 2 27 26 27 4 5 25 27 24 2 2 28 28 31 3 3 21 26 30 3 2 27 21 26 5 5 6
>854_824_731_F3
14 20 20 23 23 14 26 20 28 26 18 26 22 26 30 14 25 24 26 28 17 28 18 28 29 7 25 11 26 27 18 21
>854_1300_825_F3
32 28 27 26 24 28 30 27 26 21 24 29 26 30 29 27 30 23 28 29 26 31 26 31 29 21 25 28 19 27 23 24
>855_103_1176_F3
31 25 27 28 32 29 28 28 19 32 30 25 23 20 19 20 21 27 29 28 19 21 26 27 26 28 28 29 19 25 5 30
>855_133_1168_F3
26 32 28 25 15 31 27 29 19 28 27 27 24 24 29 29 24 25 23 28 25 25 8 27 29 25 24 25 13 25 31 8 1
...
```

Es gibt
- Kommentarzeilen (mit #)
- Kopfzeilen (mit >)
- Datenzeilen (Zahlen)

Wir wollen nur die Qualitätswerte (670000 x 35 – Matrix) extrahieren.

Erzeugung einer Datei mit ausschließlich Qualitätswerten

```
% grep -v '[#>]' reads.qual > q.txt
% head q.txt
% wc q.txt                                    # 676773 Zeilen
```

Die Option -v bei grep invertiert die Logik und gibt die Zeilen aus,
die die Bedingung, # oder >zu enthalten, nicht erfüllen:

```
25 27 27 2 29 30 25 3 2 27 26 27 4 5 25 27 24 2 2 28 28 31 3 3 21 26 30 3 2 27 21 26 5 5 6
14 20 20 23 23 14 26 20 28 26 18 26 22 26 30 14 25 24 26 28 17 28 18 28 29 7 25 11 26 27 18 21
32 28 27 26 24 28 30 27 26 21 24 29 26 30 29 27 30 23 28 29 26 31 26 31 29 21 25 28 19 27 23 24
31 25 27 28 32 29 28 28 19 32 30 25 23 20 19 20 21 27 29 28 19 21 26 27 26 28 28 29 19 25 5 30
26 32 28 25 15 31 27 29 19 28 27 27 24 24 29 29 24 25 23 28 25 25 8 27 29 25 24 25 13 25 31 8 1
29 24 23 27 24 5 8 25 24 32 26 18 27 21 28 23 26 27 22 20 24 21 19 15 25 27 18 24 8 26 24 18 24
28 30 17 32 29 24 28 28 23 24 30 30 26 24 22 27 23 26 21 22 23 25 27 11 18 29 23 27 4 16 28 24
```

Einlesen in R

```
% R                              # Shell: Starten von R

> q = read.table('q.txt')       # R: Einlesen der Datei

> dim(q)                        # Orientierung: Größe?
[1] 676773     35

> colnames(q)                   # Namen der 35 Spalten: automatisch
 [1] "V1"  "V2"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10" "V11"
[13] "V13" "V14" "V15" "V16" "V17" "V18" "V19" "V20" "V21" "V22" "V23"
[25] "V25" "V26" "V27" "V28" "V29" "V30" "V31" "V32" "V33" "V34" "V35"
```

Mittlere Qualität und Variabilität jeder Position

```
> m = mean(q)
> s = sd(q)

> plot(m, col="red", type="o")    # Plot der Mittelwerte

> X11()
> plot(s, pch="+", col="blue", type="o")
```

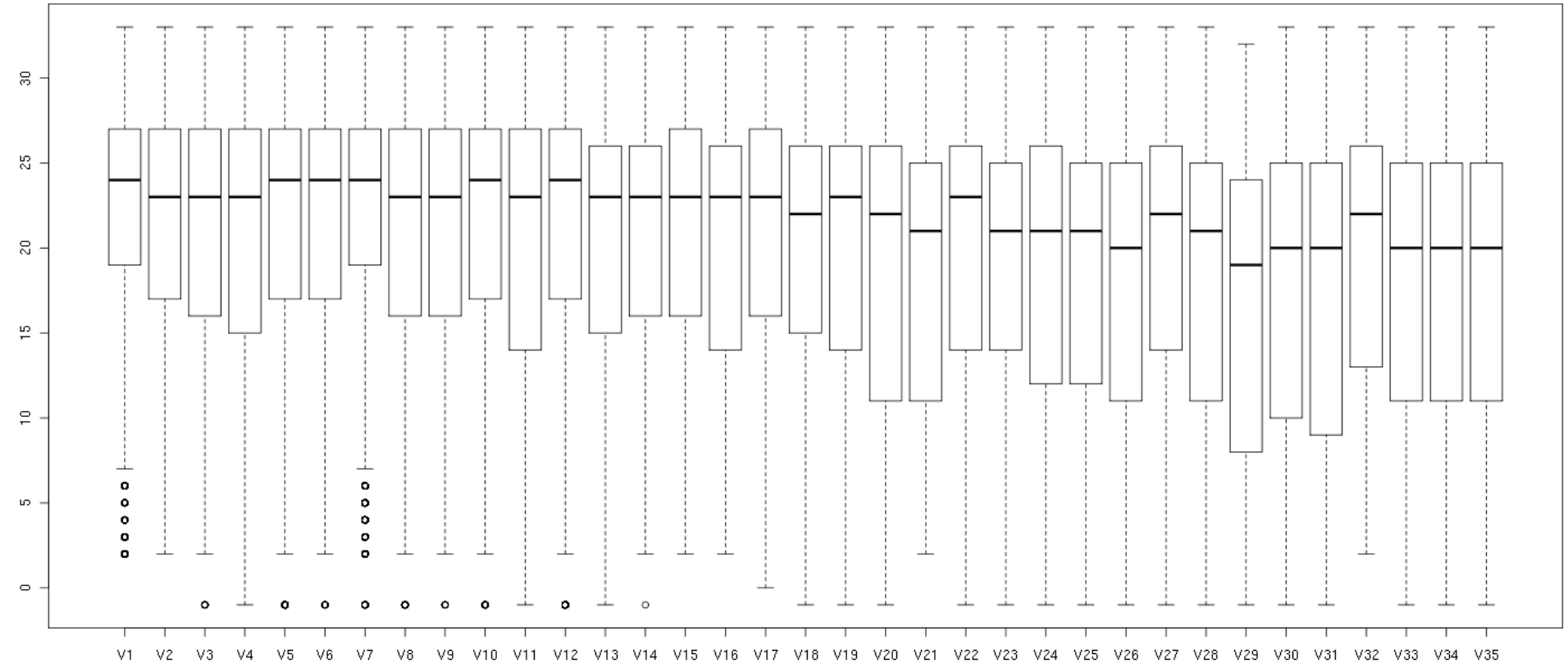
Merke

- Mit X11() kann man ein weiteres Plot-Fenster öffnen.
- Die einfachen beschreibenden Statistiken (wie `mean`, `sd`) arbeiten spaltenweise auf Tabellen (dataframes)!
- Das funktioniert leider nicht mit `median`, `IQR`.
- Aber mit `boxplot`...

Boxplot der Qualität jeder Position

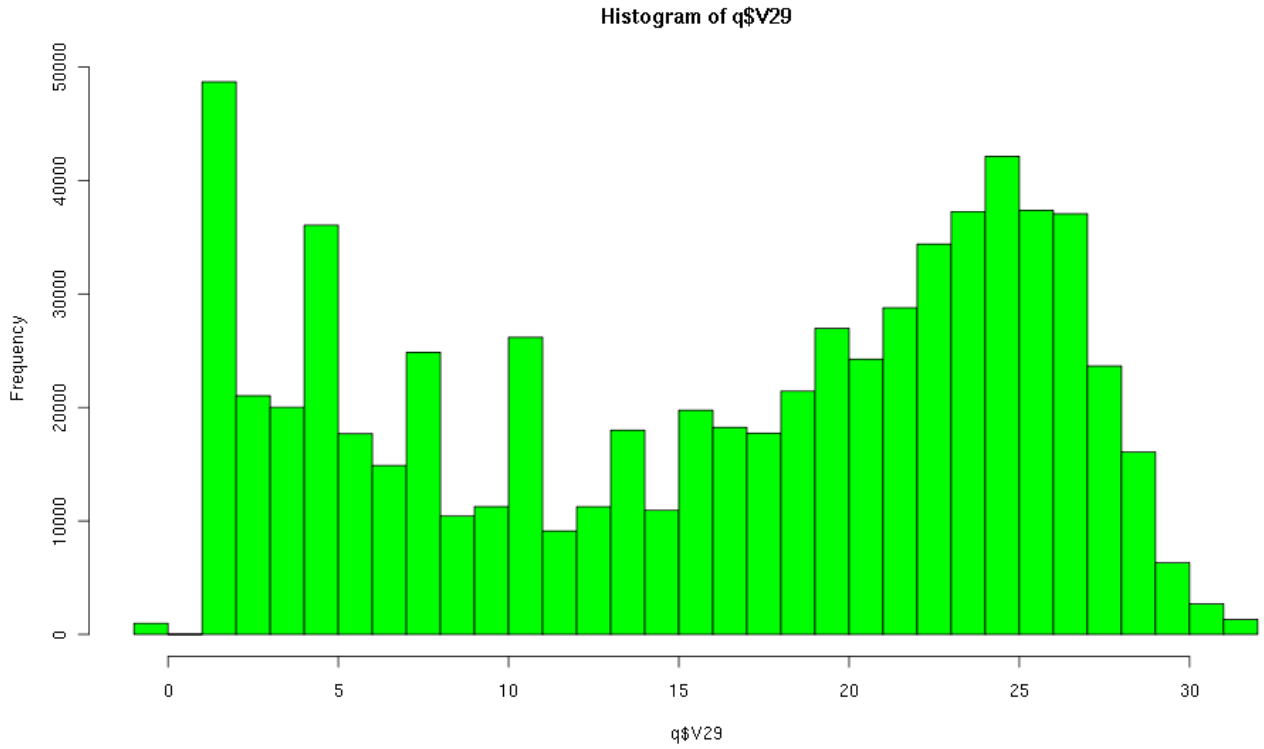
```
> boxplot(q)          # dauert...
```

Position 29 fällt auf.



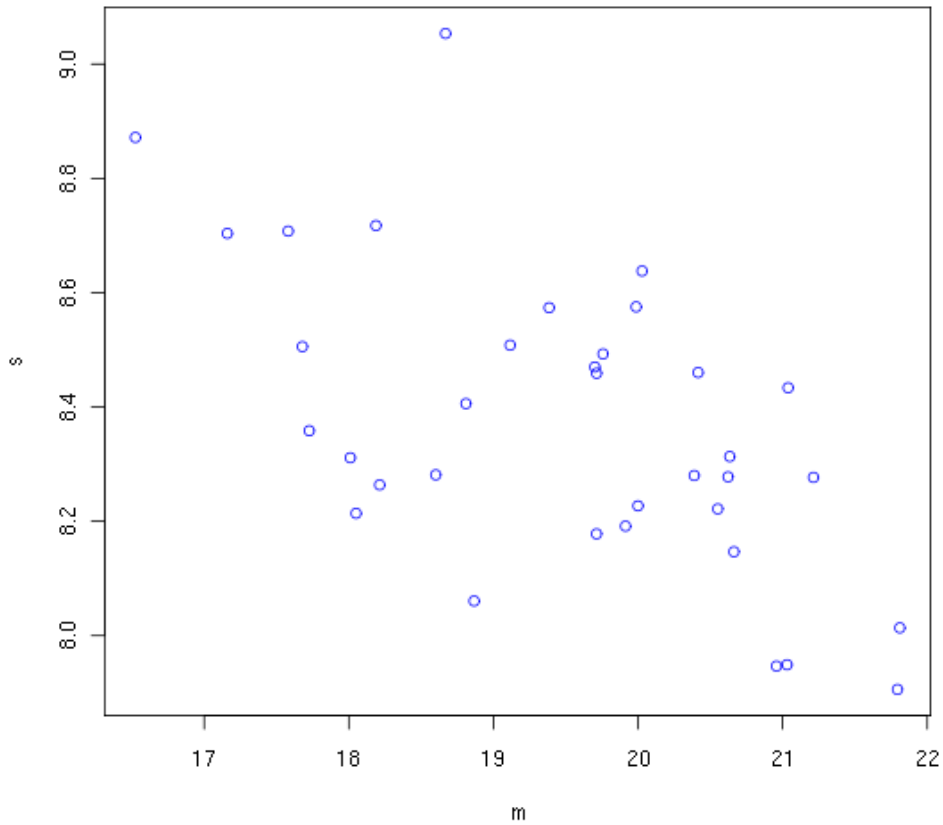
Qualitätshistogram (Position 29)

```
> attach(q)                      # spart Tipparbeit
> min(V29)                       # min. Qualität an Position 29
> max(V29)                       # max. Qualität an Position 29
> hist(V29,35,col="green")       # Histogramm
```



Korrelieren Mittelwert und Standardabweichung?

```
> plot(m,s, col="blue")     # Scatterplot m gegen s
> cor(m,s)                  # Tendenz?
```



Wichtige Verteilungen und ihre Eigenschaften

Stetige Verteilungen

norm: Normalverteilung
unif: Gleichverteilung (Uniform-Verteilung)
exp: Exponentialverteilung

Diskrete Verteilungen

binom: Binomialverteilung
pois: Poisson-Verteilung

Funktionen auf Verteilungen

d... Dichte
p... kumulative Verteilungsfunktion
q... Quantile
r... Zufallszahlen

Dichte und kumulative Wahrscheinlichkeiten

Dichte d____(x)
gibt an, wie wahrscheinlich Werte aus der Verteilung in einem Bereich liegen
W-keit, dass zufälliger Wert X (aus Verteilung) in]a,b] liegt, ist:
Integral von a bis b über die Dichte
z.B. dnorm(x) – Dichte der Standard-Normalverteilung an der Stelle x

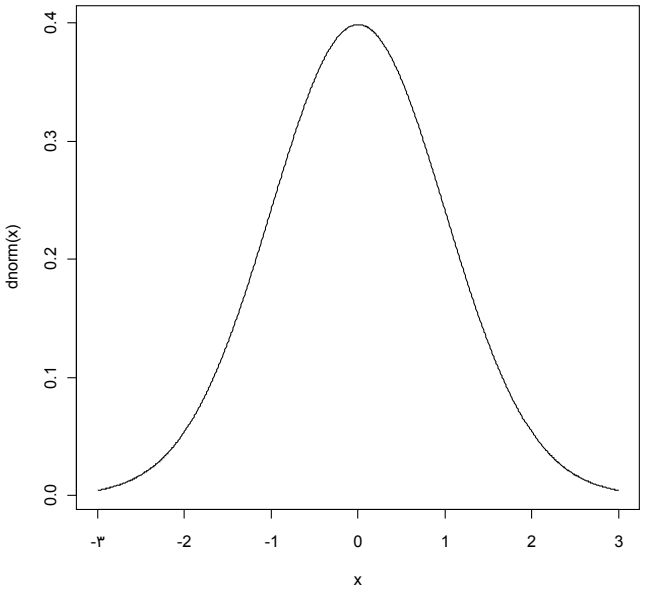
kumulative Wahrscheinlichkeitsfunktion p____(q)
Integral (von minus unendlich) bis q über die Dichte
W-keit, dass zufälliger Wert (aus Verteilung) $\leq$ q ist
z.B. pnorm(x) – Wahrscheinlichkeit, dass normalverteilte Zahl $\leq$ x ist

Beispiel Normalverteilung (Gauss-Verteilung)

Charakterisiert durch (Mittelwert, Varianz), z.B. bei Fehlern in Messdaten.
Typisch: Geringe W-keit, stark vom Mittelwert abzuweichen („fällt schnell“)

Dichte (dnorm) plotten

```
x = seq(-3,3,by=1/128)
plot(x,dnorm(x), type="l")
```



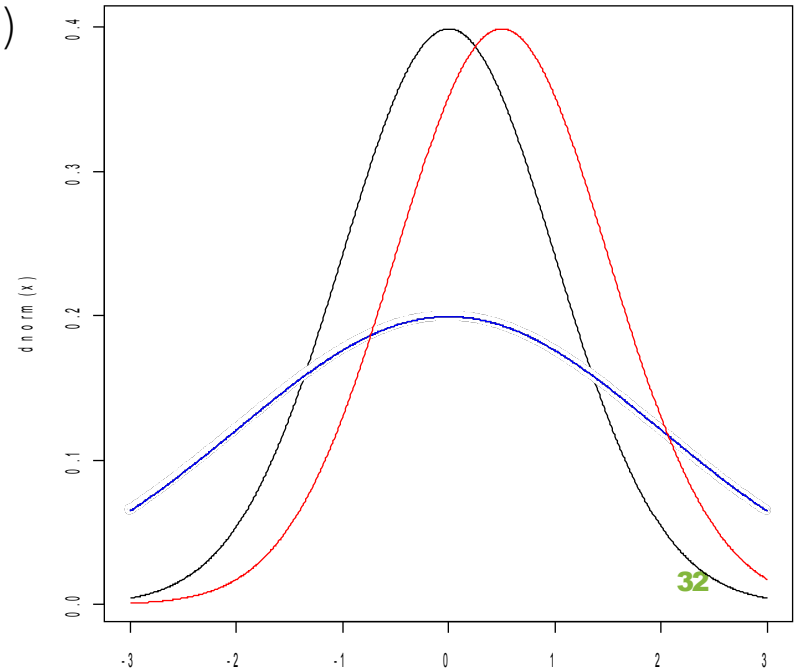
Beispiel Normalverteilung (Gauss-Verteilung)

Charakteristisch für die Normalverteilung
- Symmetrie um den Punkt maximaler Dichte: Median = Mittelwert
- Konzentration der Dichte um den Erwarungswert („fällt schnell“)

Zwei Parameter
- Mittelwert: beschreibt Lage
- Standardabweichung: beschreibt Skala
  (Varianz: Quadrat der Standardabweichung)

Standardnormalverteilung
Mittelwert 0
Standardabweichung 1

Parameter kann man übergeben an
`dnorm(mean=..., sd=...)`.



Quantile

Quantile von Verteilungen
Bisher: Quantie aus Daten
p-Quantil: Wert x, so dass Anteil p der Daten <= x ist
Jetzt: Quantile für Verteilungen
p-Quantil: Wert x, so dass W-keit p besteht, dass zufällige Zahl <=x ist.

Beispiel

```
qnorm(0.5) = 0    # Median liegt bei 0
```

Verschiebung und Skalierung
Stammt x aus Standardnormalverteilung (mean=0, sd=1)
so verhält sich ax+b wie aus Normalverteilung mit sd=a, mean=b.

Quantil-Quantil-Plot

Stammen zwei Datensätze aus der gleichen Verteilung?
Wenn ja, müssen auch die Quantile gleich sein.
Wenn die Verteilungen sich nur in Lage (mean) und Skala (sd) unterscheiden, sind auch die Quantile entsprechend skaliert.

qq-Plot
plottet Quantile einer Verteilung gegen Quantile anderer Verteilung.

gleiche Verteilung:	Gerade mit Steigung 1 durch 0
verschobene Verteilung:	Gerade mit Steigung 1 durch ??
skalierte Verteilung:	Gerade mit Steigung ?? durch 0
verschobene skalierte Verteilung:	Gerade
andere Verteilung:	keine Gerade!

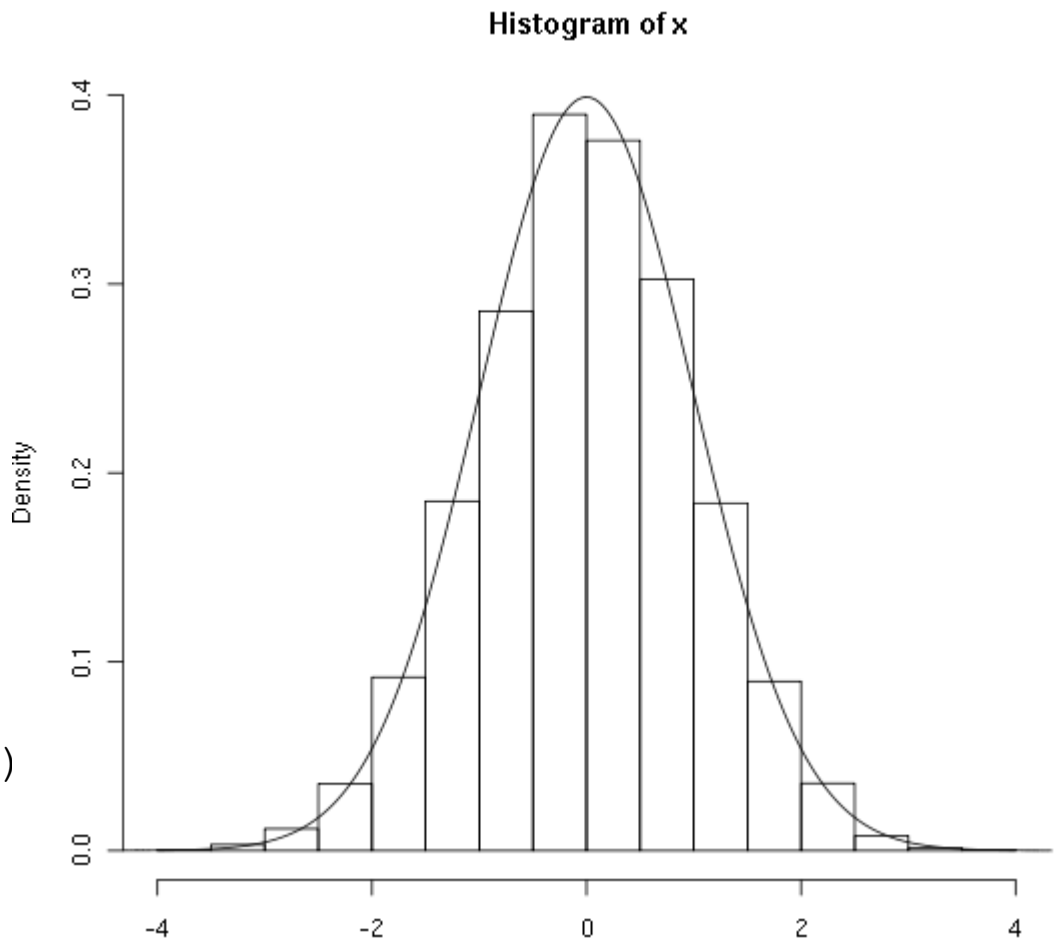
Zufallszahlen

Ziehe n Zufallszahlen
Aufruf: r____(n);
evtl. weitere Parameter angeben
Beispiel

```
x=rnorm(100, mean=5,sd=3)
```

Histogramm vieler Zufallszahlen rekonstruiert Dichte

```
x = rnorm(10000)
hist(x)
z = seq(-5,5,1/128)
points(z,dnorm(z),type="l")
```



qq-Plots mit Zufallszahlen aus der gleichen Verteilung

Experiment

Ziehe 2x aus der gleichen Verteilung viele Zufallszahlen.
(Diese sind nun verschieden, aber gleich verteilt.)

qq-Plot muss eine Gerade sein.
Starke Krümmung bei ganz verschiedenen Verteilungen.

```
> x = rnorm(100)
> y = rnorm(100)
> qqplot(x,y)
# in etwa eine Gerade
```

```
> x = rexp(100)
> y = rnorm(100)
> qqplot(x,y)
# stark gekrümmt
```

Wichtiger Fall: Test auf Normalität

Gegeben: Daten (Vektor) x
Frage: Stammt x aus einer Normalverteilung?

Kann Folgendes tun:

```
y=rnorm(10000)
qqplot(x,y)   # Gerade?
```

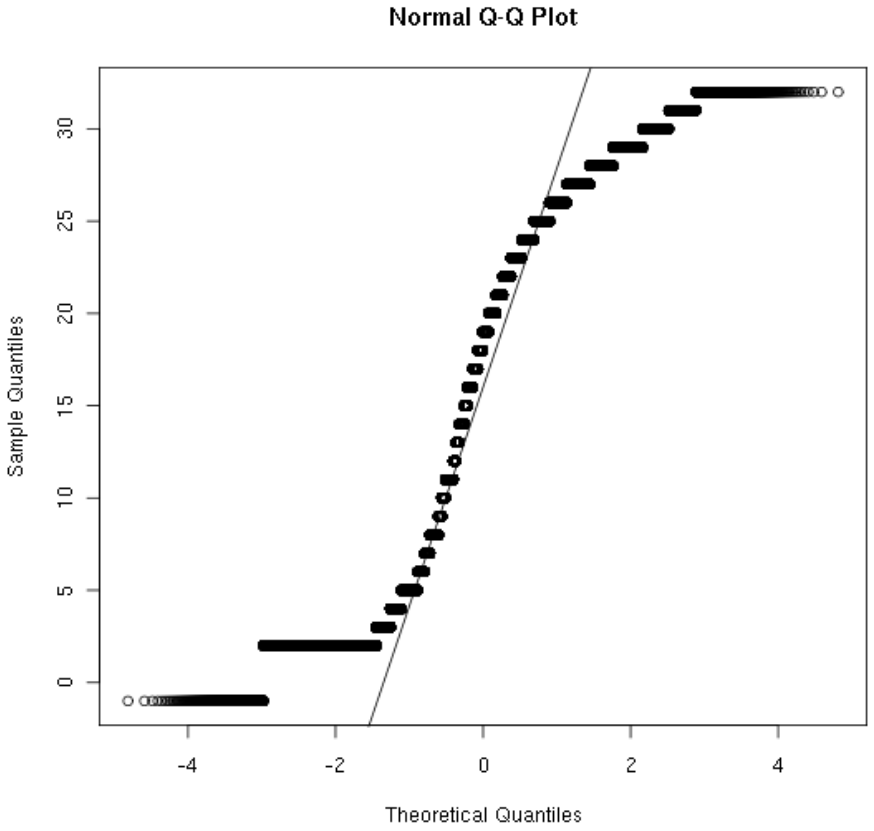
Einfacher:
keine Zufallszahlen ziehen, sondern mit theoretischen Quantilen vergleichen

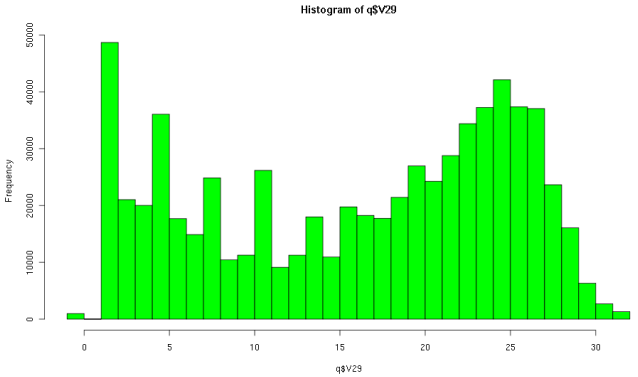
```
qqnorm(x)     # Gerade ??
qqline(x)     # Beste Gerade dazu einzeichnen!
```

Beispiel:
Sequenzierdaten; Qualitätswerte an Position 29 normalverteilt?

Test auf Normalität mit qqnorm()

```
qqnorm(q$V29) # Gerade??
qqline(q$V29) # beste Gerade dazu
```





Warum nicht Histogramm mit Dichte dnorm() vergleichen?

- qqplot zeigt Abweichungen besser; invariant bei Verschiebung, Skalierung

Vergleich von normalverteilten Stichproben (Vektoren)

Fragestellung:
Gegeben zwei Datensätze (Vektoren),
haben die Verteilungen, aus denen sie stammen, gleichen Mittelwert?

Erste Idee:
Vergleiche empirische Mittelwerte (Durchschnitte) der beiden Vektoren.

Problem:
Durchschnitte sind nie exakt gleich!
(Auch nicht wenn aus exakt der gleichen Verteilung gezogen wurde.)

Experiment:
Ziehe zweimal je 10 Werte aus Standard-Normalverteilung (rnorm(10)).
Berechne Durchschnitt (mean).
Berechne Differenz der Durchschnitte. Ungleich Null!

Also:
Gewisse (kleine) Unterschiede des Mittelwerts zwischen Stichproben
aus derselben Verteilung sind kein Indiz für verschiedene Mittelwerte!

Test auf gleichen Mittelwert (bei Normalverteilung)

Nullhypothese:
Stichproben stammen aus Normalverteilungen mit gleichem Mittelwert.

Alternative:
Stichproben stammen aus Normalverteilungen mit verschiedenen Mittelwerten.

Frage, die die Statistik beantworten kann:
Angenommen, die Nullhypothese trifft zu.
Wie wahrscheinlich ist es, dass sich die beobachteten Mittelwerte
um mindestens so viel wie die beobachtete Differenz unterscheiden?
Diese Wahrscheinlichkeit nennt man p-value (p-Wert).
(Für nicht normalverteilte Daten wird hier nichts ausgesagt!)

Anwendung:
Man gibt eine Grenze (Signifikanzniveau) vor (z.B. 0.05 oder 0.01).
Ist der p-Wert ≤ 0.05, sagt man: „Der Unterschied ist signifikant".
Ist der p-Wert ≤ 0.01, sagt man: „Der Unterschied ist hoch signifikant".
(Wahrscheinlichkeit für einen so großen Unterschied ist bei Nullhypothese klein!)

Der t-Test

Vergleich von zwei normalverteilten Stichproben x, y heißt
Zwei-Stichproben-t-Test.

Man „darf" diesen Test nur auf approximativ normalverteilte Daten anwenden.
(Wenn man das nicht beachtet, ist das Ergebnis bedeutungslos.)

In R:
Zuerst x,y auf Normalverteilung prüfen (z.B. mit qqnorm).
Dann: `t.test(x,y)`

…liefert viele Informationen; wichtig ist der p-value.

(Aufgabe der mathematischen Statistik und Wahrscheinlichkeitsrechnung
ist Erfinden solcher Tests und exakte Berechnung der p-values.)

Der t-Test bei Microarrays (multiples Testen)

Situation:
Microarray-Experimente von 10 Tumor-Proben und 100 Kontroll-Proben liegen vor.
Betrachte ein bestimmtes Gen.
Betrachte Genexpression in beiden Klassen (10 und 100 Werte).
Gibt es einen signifikanten Unterschied zwischen den Klassen?

Lösung:
t-Test (sofern die 10 und 100 Werte normalverteilt sind).

Aber:
Wahrscheinlichkeit, dass ein Gen-p-Wert $\leq x$ ist,
obwohl kein Unterschied besteht, ist x (nach Definition des p-Werts).
Wir testen viele Gene (~25,000).
Da ist „durch Zufall" ein p-Wert $\leq 1/25000$.
Das bedeutet noch nichts.

Korrektur der p-Werte für multiples Testen (Bonferroni-Korrektur):
Multipliziere p-Werte mit Anzahl der Gene.
Betrachte dann p-Werte ≤ 0.05 (signifikant) bzw. 0.01 (hoch signifikant).